IIT/K COMPUTER NETWORK—MICRO 78 HARDWARE INTERFACES

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By

DEBASIS DAS

to the
DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY, 1979

CERTIFICATE

This is to certify that the work on "IIT/K COMPUTER NETWORK- MICRO 78 HARDWARE INTERFACES" has been carried out under our supervision and guidance and that this has not been submitted elsewhere for a degree.

(A.S. SETHI)
Lecturer
Computer Science,
Indian Institute of Technology
Kanpur

(V. RAJARAMAN)
Professor
Electrical Engineering
Indian Institute of Technology
Kanpur

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Dr. V. Rajaraman for his invaluable suggestions and guidance. My sincere gratitude to Dr. A.S. Sethi for his constant encouragement, guidance and understanding of the practical difficulties.

I wish to thank ACES Research Engineers Mr. Arjun Raman, Mr. K.S. Ananthakrishnan, Mr. S. Manoharan and Mr. Ramanna without whose little helps hardware development might have become a nightmare. My thanks are due specially to my colleagues N.S. Narayanan and C.V. Krishna who were ideal teammates during development of the crucial line control and logic for the interfaces.

Thanks are due to Pawagi, Jain, Prabhakar, Parmeshwar for little helps and suggestions and above all to keep me going through the grind of actual hardware development.

I thank Mr. J.S. Rawat for the superb job of typing he has done.

July, 1979 - DEBASIS DAS

ABSTRACT

A system is proposed to link up DEC 1090, TDC-316 and IBM 1800 at the IIT/K Computer Centre. These three machines will be connected in a star configuration with a micro 78 microcomputer acting as the central switching node. The communication link are full duplex with a data rate of 10 K bits/sec. The network is designed for resource sharing and for experimental work in various aspects of computer networks. Flexibility is of prime concern in such a system. This report describes the overall system design, line protocol design and the hardware implementation of the line protocols at the switching node, the micro-78.

CONTENTS

CHAPTER 1

INTRODUCTION

A computer network is a varied and complex collection of computing resources. The network makes all these resources available to any user irrespective of the actual geographical area where these resources may be located [KAHN-72]. The resources could be as diverse as data bases, programs or special computing hardware like ILLIAC-IV for example.

Advanced Research Projects Agency (ARPA) of the U.S. Deptt. of Defence was the poineer in networking of computing facilities. Their network ARPANET became operational in the late sixties. This was aimed at reliable sharing of speciallized computing resources [KLEI]. Other examples of operational networks are SITA [BRAN-72] for airlines reservations, National Physical Laboratory's (U.K.) network [DAVI-68] and ALOHA [ABRA and KUO], the University of Hawaii's network which uses a unique technique of communication.

Applications of networks are varied, resource sharing being the most important - immediately followed by load sharing of computing facilities.

The GE and TIMSHARE networks are examples of making time shared access available to all design engineers over widely

distributed geographical areas. Special purpose networks for credit checking and electronic fund transfers in banking transactions are quite common. Police networks with small terminals in patrol vehicles for instant check on criminals is another example of this variety.

1.1 Basic Concepts:

A network can be viewed as a set of nodes interconnected via a set of communication channels. These nodes contain the processing resources of the network called HOSTS [WALD-75]. Since the communication overhead could be large, it is more usual to have interface processors handling communication tasks alone, catering to a number of hosts at a geographical location. These processors along with the communication channels form the "communication subnetwork". For all practical purposes this subnetwork is transparent to the users so that he need not be unduly worried about the detailed mechanisms of information being carried back and forth.

The units of information transmitted between nodes are called messages. Unlike the normal telephone messages, where a physical channel is set up between users, these messages are routed from node to node, each node deciding the next hop to another node so as to reach the destination ultimately.

One reason for adopting this kind of switching is to avoid large call set-up delays as is usual in circuit switching. Another major reason is that switching of a message is much more adaptive to changes in circuit conditions. This then can be made to choose a low delay path all the time. To achieve this further, messages are broken up into packets and each packet is handled as a unit. This is called packet switching [CROW-75] . The interface processors now have an extra job of creating packets at the sending node, and assembling packets, which may be received out of order, back into recognizable messages at the destination node. The need for processors to handle these and other communication oriented overhead like code conversion etc. becomes apparent.

The way these nodes are interconnected has an important effect on the operation of the network in terms of its simplicity, reliability, delay targets achieved, cost etc. Some of the possible configurations are star, data loop, tree, fully connected, distributed or hierarchical. One has to choose among these configurations consistent with the design goals. Of these the star, loop and distributed configurations are the prime contenders for a small local network KAHN 72 . A detailed discussion on these and other configurations follows in Chapter 2.

1.2 Protocols and their functions:

Protocol is defined as "the customs and regulations dealing with diplomatic formality, precedence and etiquette". This regulates orderly exchange of information between states. Likewise an orderly interchange of information between processors requires a set of protocols. Protocols are generally designed in a layered fashion for ease of implementation as well as implementing later changes without affecting the lower level protocols. At lower levels, the protocol is more concerned with details of communication mechanisms, peculiarities of the medium etc. The lowest among them is the line control protocol. Higher level protocols are more function oriented [FRAS 76].

Line control protocols take care of the actual details of communications between two processors connected via a physical line. At this level maintenance of synchronization between sender and receiver, initiation of message transfer, termination of message transfer, acknowledgement of a message received as well as the error detection and recovery [GRAY 72] are important.

1.3 IIT/K Computer Network:

In spite of rapid progress in computer networking in other countries, nothing much has taken place in this country.

Save for an experimental link-up of TDC-316 computers to DEC10 system at TIFR, Bombay there are no operational computer nets in this country although some are in the planning stages. Among these are AIR-INDIA's reservation system and ISRO's link-up of their computers at different centres.

Setting up of a network of computers at the IIT/K Computer Centre would provide practical experience needed in the design and implementation of a network. This would also serve as a test bed for conducting experimental studies in various aspects of networks like flow control, routing algorithms, protocols etc. which are essential to gain first hand knowledge in these areas.

To the general user such a network should be able to provide some practical facilities too. Sharing of peripherals would be one of the most useful among these. For example the IBM 1800 card punch could easily become available to a user of DEC system which does not have this facility. A TDC 316 user could possibly use the tape-drives of IBM 1800 or DEC system convert the 7 track format of 1800's tape to its own 9 track tape drives. TDC 316 will be able to act as a Remote Job Entry (RJE) station to DEC system. Using it as a concentrator for terminals, thereby allowing system expansion is also on the boards.

Both of the above two types of use require that implementation of the basic hardware be simple and flexible. This then would allow later implementation of software and experiments without being limited by the capability of hardware. Economy is another absolute necessity as funds are limited. The star configuration with Micro-78 as the switching node was decided on with these goals in view. This thesis describes the design and implementation of the Micro-78 interfaces. Companion theses CVK 79, NSN 79 describe the other relevant parts.

Chapter 2 explains the system configuration, the design goals etc. while Chapter 3 discusses the development of a suitable line control procedure. In Chapters 4,5 and 6 hardware design and development is discussed. Chapter 7 is based on what all has been achieved and what all could further be achieved starting there.

CHAPTER 2

SYSTEM DESCRIPTION

Like in any system design, in network design also, on the one hand we have design goals, on the other constraints imposed by the network structure chosen. Various trade-off are possible in the situation. This chapter elaborates on the goals for the proposed network and the various constraints due to particular design choices.

The whole process of designing a network may be broken down into the following broad logical steps.

1. Definition of aims and applications.
2. Selecting a network configuration in view of (1).
3. Design of the lowest level protocol, the line control protocol, to support data communication.
4. Estimation of hardware complexity of the interface circuitry to support (1),(2) and (3).
5. Decisions on trade-offs at steps 2,3 and 4 to achieve the original goals (step 1). This requires a number of iterations through step 2,3 and 4.
6. Design of the software interface to let the nodes communicate.

2.1 Goals:

The network under discussion is going to be used 'n an academic environment~~s~~. Flexibility, as such, becomes the prime goal to achieve enugh flexibility for future changes, as well as facilities to support experiments to be conducted on this network is of vital importance. Hardware design has to be such as to impose minimum limitations on such work

The foregoing dictate that the hardware interfaces be

(a) simple and general

(b) at least logically similar, even though machine pecularities are to be taken care of

(c) useful, in the sense that users be able to access system facilities and peripherals of any of the nodes in the system.

2.2 Network Configurations:

Various configurations are possible in a computer network. They are as follows:

i) Star - (a) With a central computing facility, users at the end of various radial lines.

(b) With a no. of computing m/c connected to a central switching node.

ii) Loop(or Ring) - No. of nodes connected through a loop of synchronized one way highway.

iii) Tree - A layered connection of nodes to multiplexers then onto a set of users for each of these multiplexers.

iv) Fully connected - Each of the nodes being connected to every other node in the network.

v) Distributed - Similar to above but only selective interconnections are made.

vi) Hierarchical - In other words a network of networks.

Of these only the (i) b , (ii) or (iv) configuration are suitable for a small local network like the present one being envisaged.

A major disadvantage of the star network is that it may require large line lengths between central switch and each of the hosts. In our case, these are well within tolerable limits, all the three systems being under the same roof. Reliability of the network is mainly dependent on the central switch and thus quite low. But the simplicity and low cost achieved could well justify the choice of this configuration.

On the other hand, the loop configuration is a tried out one. Universities (such as at IRVINE, USA) have such networks operational. Line control procedures and hence the

hardware interfaces are quite complex. Designing a flexible system with this configuration is a fairly complex task with resultant high cost.

Fully or partially connected networks score over the previous two in terms of reliability. But the necessary high extra cost and hardware rule out their usefulness for the present exercise. The proposed network thus has the star configuration shown in Fig. 2.1.

The choice of the central switching node is the next step. A micro 78 microcomputer was chosen. This avoids designing a special purpose hardware switch. Other major consideration was to study the usefulness of a microcomputer in such a role.

Having made the above decisions, the choice of full - duplex serial transmission is almost automatic in view of our goals already discussed.

2.3 Data Transfer Modes & Data Rates (Fig. 2.2):

Data transfer from host memories onto the serial line can be accomplished in either of two ways

(1) DMA

(2) Through interrupts

In the DMA node the interface gets/dumps data from/to memory via direct access whenever the need arises. In this

mode it is possible to achieve high transfer rates without slowing down the processor appreciabily. But to accomplish this the DMA controller, of necessity has to be quite complex. It has to keep track of addresses, no.of transfers made etc., and thus the hardware becomes quite complex.

In the interrupt transfer mode the processor takes on the responsibility of the house keeping as regards the transfer. This does tie down the processor. The interfaces have only to generate interrupt signals whenever the need for a transfer arises. This does not impose any extra complexity on top of the basic hardware requirement to support communication.

The deciding factor then is whether the slowing down of the processors due to data transfer overhead is tolerable without incurring extra cost for DMA control.

Data rates of 9.6Kb/sec being standard, the same was adopted. Assuming that along with an 8 bit byte a parity bit is also sent, total serial transmission time of 900 sec is required at the rate of 10 Kb /sec. which is the data rate supported by the network. Thus any interface generates an interrupt every 900 sec. The most crucial point is that of micro 78 which handles six interfaces (Fig. 2.1 and Fig. 2.2) in the final form. On an average then it expects an interrupt

every 150 usec during which time it can execute about 75 instructions, sufficient for a fair amount of software to handle these interrupts

2.4 Overview of the System:

The following features then emerge as the specifications of the total system.

1. Star configuration with micro 78 as the switch.
2. Full duplex data transfer at 10 Kbits/sec rate.
3. Interrupt mode transfer of data between interfaces and host processors except for IBM 1800. This has data channel transfer mode, somewhat like a DMA transfer, available.

Discussion on the design of an efficient line protocol is deferred to the next chapter. The following section discusses the salient features of micro-78 the microcomputer whose interface design is the topic of this dissertation.

2.5 The Switching Computer:

This is a microcomputer based on the INTEL's 8080 microprocessor. The configuration in which it will be connected to the system would support a TTY, 24 K bytes of semiconductor memory for supervisory and interrupt handling programs as well as the store and forward message buffers,

console with display and switches for program loading, a real time clock with a programmable timer and a floppy disk drive. The TTY has paper tape reader and punch which could be used for preparing and loading large pieces of software and then made resident in the floppy disk.

This microcomputer has a word length of 8 bits. Instructions could be multibyte (upto 3 bytes long). Aside from the accumulator it has six other general purpose registers. These could be used in (B-C, D-E, H-L) pairs to hold 16 bit addresses or data. Normal memory accesses are indirectly via H-L pair. Other pairs also could hold memory addresses for certain instructions.

I/O is through a set of 256 ports, each of which can be used as either input or output. When executing I/O instructions both the upper and lower bytes on the address lines of the bus carry I/O port address information. Other mode of I/O available are through DMA and interrupt.

DMA is handled through two control lines DMAR and DMAA ack. DMAA ack line is actually two DMA ack in and DMA Ack out. This facilitates daisy chaining of DMA devices.

Interrupts are handled through a single interrupt line. A request on this line causes program control to

branch to a location 70_8. Any interrupt handling program has to start at this location in memory.

Another form of I/O is possible with micro-78, which has been adopted, This is to treat the device registers as memory locations. This form of I/O gives the added flexibility of treating the interface registers as memory locations and operating on them as such. The communication is via the address and data bus. The control signals are READ/WRITE (TSRM & WXM respectively) and SSYN. As soon as Read/Write line becomes active, SSYN line must be activated. This should remain active till sufficient time has elapsed to get the data IN/OUT of the hardware registers. Fig. 2.3 gives the mother board configuration viewed with micro-78 back panel open.

TABLE I: Pin nos. of the bus signals

SIGNAL	PIN No.
Address bus	A1 to A16
Data bus	A17 to A24
$\overline{\text{TSRM}}$	A 25
$\overline{\text{WRM}}$	A 26
$\overline{\text{SSYN}}$	A 27
$\overline{\text{MC}}$	A 28
$\overline{\text{TSRI}}$	A 29
WRO	A 30
$\overline{\text{IR}}$	A 31
$\overline{\text{DMAR}}$	A 32
DMA ACK out	A 33
DMA ACK in	A 34
+5V	A 39
GND	A 40

NOTE: Only A connector pin out is shown as that is the point of interface with rest of the hardware.

All signals low active. Logic levels: 1 - TTL "0"
0 - TTL "1"

Real time clock & programmable timer:

The real time clock provides time of the day in hours, minutes and seconds. This clock is monitored by a 'watch dog' to provide enterrupts on failure of this clock. Interrupt is also generated on overflow.

The programmable timer provides interrupts at the end of programmed interval. The following things can be programmed.

(a) A count, which decides after how many 'ticks' timer overflows.

(b) 'tick' rate of 10 Hz, 1 KHz, 10 KHz or 1 MHz.

(c) Whether interval timer should start counting again at the end of a counting sequence (one shot interrupt or repeated interrupts at programmed interval).

Only the relevant hardware features are described above. Details of other features as well as software features are available in micro 78 system manual. MCS-80 handbook by INTEL will be particularly helpful for software features.

How these features have shaped the hardware design is discussed in the chapters on hardware design.

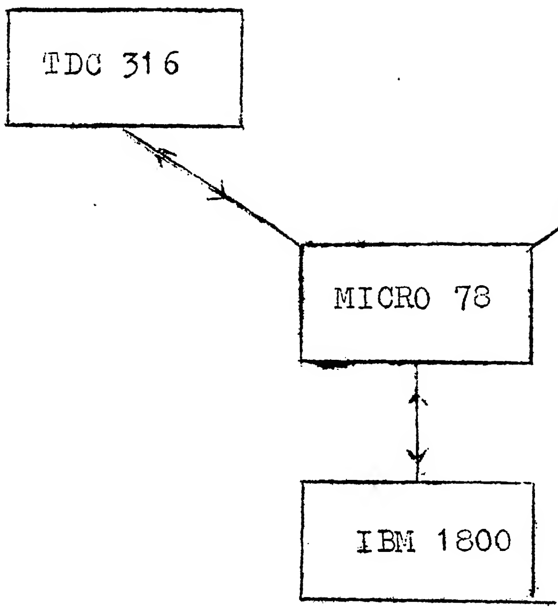


Fig. 2.1: System Configuration

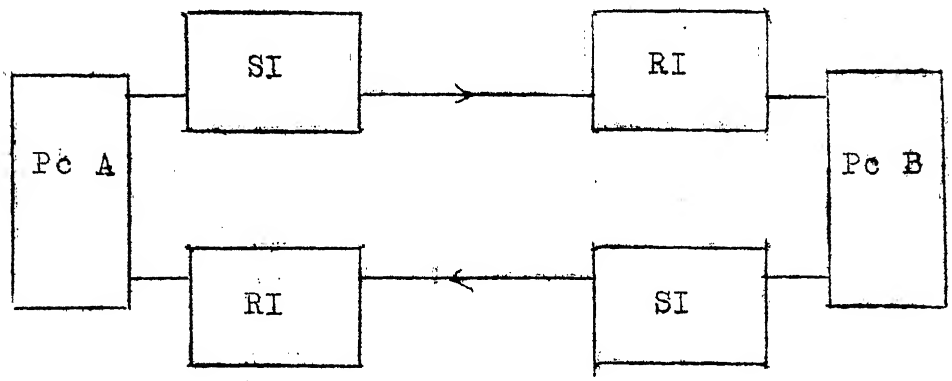
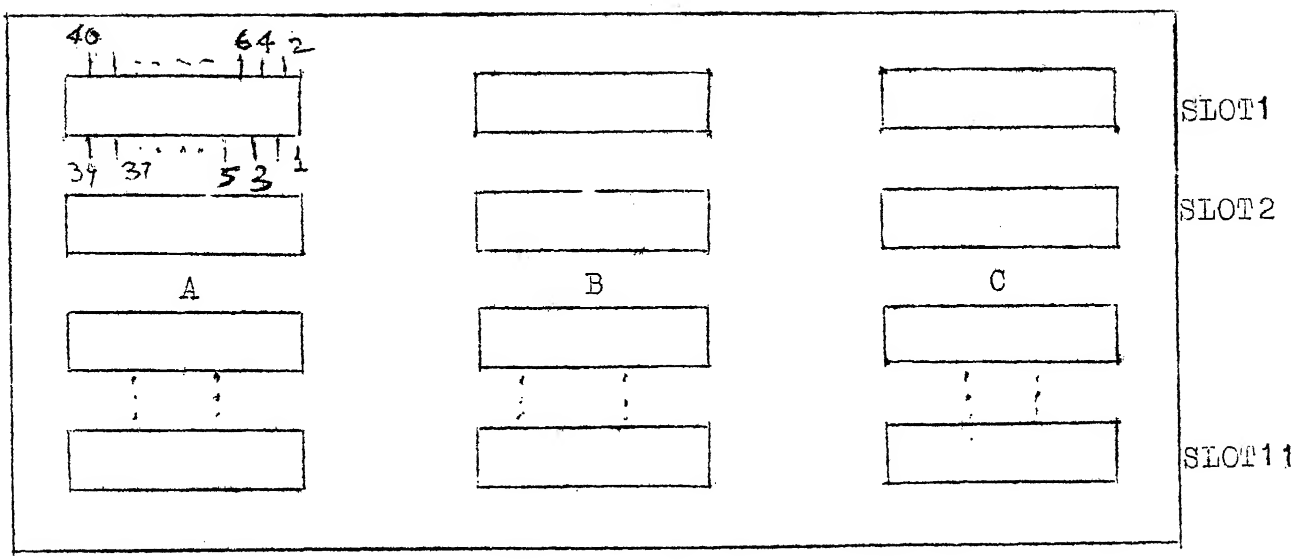


Fig. 2.2: Dual Processor Subsystem



NOTE: SLOT1; for CPU + TTY ONLY

SLOT4,5,and 6: for FLOPPY Controller

Fig. 2.3: Micro 78 Back panel

CHAPTER 3

LINE CONTROL PROCEDURES

Essential functions of a line control procedure are [GRAY 72] :

i) to establish a link

ii) to synchronize the sender and receiver

iii) to send and receive messages

iv) to recognize and correct, if possible, errors during transmission.

In the present set up dedicated links are available between corresponding senders and receivers. So the first problem is that of synchronization. This has two aspects, those of acquiring and then maintaining the synchronization. Acquiring synchronization means recovery of the transmitted clock from the random bit sequence coming in. Once this is done, the receiver should be able to mark out the points in time where a character starts/ends.

With asynchronous communication a special start bit for every character is recognized by the receiver. This marks the beginning of a character. The receiving clock, nominally at transmitter clock rate, also is started afresh at this time so that bit and byte synchronization is achieved.

In the synchronized communication mode the receiving clock must be recovered from the continuous stream of incoming bits somehow. Byte framing is facilitated by transmission of a special character SYN. When this is recognized the times for subsequent characters could easily be marked out.

Since channels are never noise free, possibility of errors has to be taken into account, and recovery procedures have to be designed as well.

3.1 Network Line Protocols:

Characters sent over the lines between processors could be either data or control characters. It is essential that they be distinguished.

On the other hand, the bit stream intended to be a sequence of data characters may contain bit combinations re ssembling control characters. At these times, if the interface react to them as it would with a control character, the communication may completely go hywire.

One way of avoiding this is to ban such data combinations thereby restricting the users. The other approach is to make data transmission completely transparent. That is interfaces respond to control characters only when they are in a state to expect such control characters.

Length of the message is also important, as this has to do with maintenance of synchronization. But with the scheme of clock recovery adopted this is not a very critical parameter and can be fixed by higher level protocols.

When a sending interface is not sending data it keeps sending a control character SYN (16 in Hex). On detection of this at the receiving interface, a flag is set and a counter started. The flag signifies byte synchronization acquired. Only when this is done proper receiving action can start. The counter counts off 9 bits from the receiving clock marking out the next byte and the parity bit. The receiving clock is generated from the incoming data bits thereby being synchronous to the transmitter clock. This is how bit and byte synchronization is done.

A message from the transmitting end must be preceded by SOM (01 in Hex). Acknowledgements are single byte messages of the character ACK (7C in Hex). When either is received proper reception starts. That is bytes, as received, are transferred into memory of the processor through interrupts. Errors are checked before putting the characters into the memory. For a proper message (other than ACK), information about the number of bytes in it carried in the header and no special End Of Message character is required.

While the line is idling any character other than SYN resets the byte synchronization flag. The same thing happens if a SYN is not recognized between two messages.

Once a scheduled message is completely sent out the processor starts waiting for a period 'TIME OUT' to enable the receiver to send an 'ACK' message. To facilitate proper identification MSB of the ACK byte carries a 0/1 sequence number. The sending processor repeats the message if no ACK is received before the time-out, else it goes for next scheduled message.

Error checking consists of checking the ninth bit which is the odd parity of the byte preceeding it. An error flag is set to indicate parity failure. With small distances to be covered in this case, it was assumed that error rate would be low enough for single bit parity checks to be sufficient. In case this flag is set during a message reception the receiver would not send back an ACK, thereby causing the last message to be repeated. Error recovery in this particular protocol essentially consists of this action alone.

Chapters 4,5 and 6 discuss the implementation of hardware that support the protocol discussed above.

CHAPTER 4

HARDWARE DESIGN OF COMMON CONTROL CARD

This and the following two chapters discuss the hardware implementation of (1) Common Control Card (CCC) (2) Sending Interface card (SI) (3) Receiving interface (RI) necessary for interfacing the Micro-78 to the IIT/K NET.

Since the line protocol as discussed in Chapter 3 does not apply to DEC 1090 exactly, the sending and receiving interfaces necessary for it will be somewhat different. The IBM 1800 and TDC-316 legs are identical. Hardware development then consists of one CCC, two SI's and two RI's.

It is desirable that the two SI's and two RI's be identical and interchangeable among their types. Without the CCC some common functions will have to be duplicated on each of these four as well as the future DEC 1090 interfaces. This increases complexity unnecessarily besides wasting chips. The common functions, if centrallized would avoid this and that was the whole reason of existence of the CCC. The common functions are discussed later in this chapter.

4.1 Data Coding Scheme:

Data coding scheme is central to the design philosophy and hence is discussed at the outset of the design. Since no

security aspect is involved, data coding has to take care of the fundamental requirement of recovery of a replica of the transmitted clock alone. The following (among many possible schemes) was adopted because of its simplicity and the ease with which data could be detected at the receiving end (Fig. 4.1).

If data is detected at mid bit (50% of clock) time it will be decoded unambigously. A further advantage of the scheme is that a transition is available at the beginning of every clock cycle. This obviates the necessity of any tracking circuit to maintain synchronization during transitionless times in the incoming bit stream. A simple monostable is triggered at every edge generating approximately 1/2 clock cycle output pulses. This generates the receiving clock.

Over short distance spans as used in the net and a crystal controlled transmitter clock, this scheme turns out to be sufficient for our purpose. Simplicity and hence high reliability is obvious. Monostable jitters are also negligible compared to the wide band ($>$ 25% but $<$ 75% of clock period of 100 s) over which data could be detected reliably. Further any change in TX clock would be easily followed by the RX clock, except that sometimes the pulse

width may have to be changed by changing the simple RC timing component.

4.2 Design of the Common Control Card:

Each of the interfaces has two processor accessible registers. (Twelve in all). For the purpose of communication with the micro-78 the addresses of these will have to be detected. Further two more locations are used for generating interrupt vectors (as discussed later). Address decoding could be simplified by blocking these addresses. This then is a function that is a candidate for CCC.

Any communication between the m/c and the interfaces needs a particular handshake whereby the interfaces have to generate a synchronizing signal (SSYN) and keep that active for the duration of read/write. This then is another function that could be centralized.

One major issue is that of determining the source of interrupt from among the six contenders, there being a single interrupt request (IR) line input in Micro-78.

There are two possible ways of resolving this.

1. Software Poll: CPU interrogates particular status bits in all of the interfaces to determine the source. Some prioritizing of the sources is inherent.

2. Interrupt Vectoring: Program control jumps to the required service routine directly depending on the source of interrupt.

The first approach is time consuming and the Micro-78 could least offord it. Any non serviced request would give rise to (1) overlayed data in a SI (2) data overrun in a RI. which are error conditions. Second approach is the one to be adopted then.

This then is yet another function to be handled by CCC. The following three section discuss the implementation of each of these in turn.

4.3 Address Decoding:

Address decoding is shown in Fig. 4.2. Memory mapped addresses are chosen for the fourteen addressable registers in the accessible registers in the interfaces. This facilitates treating these exactly as memory locations; hence data manipulation can be done right at the register before moving it into main memory. So as not to interfere with active memory (24 K bytes for the m/c proposed) and also to simplify address decoding the topmost sixteen locations were chosen. Any register then has the address in Hex as FFFX, where X is any Hex digit. Two of them not being used. Detecting the lowermost four bits plus the

condition that other twelve higher address bits are 1's should suffice.

4.4 Generating Synchronizing Signal:

Communication between CPU and a memory location is accomplished as follows. CPU puts out the address bit combination on the address bus followed by a read memory (TSRM) or a write memory (WRM) signal as the case may be. Response of the memory system should be to activate the synchronization line (SSYN) as soon as TSRM/WRM is activated. SSYN remains activated for the duration (min. 0.5 μs) necessary to accomplish read/write and then deactivated.

SSYN is generated by the CCC whenever FFFX address is detected together with a TSRM/WRM pulse. 800 ns wide pulse was found to be sufficient. The scheme for generating this signal is shown in Fig. 4.3.

4.5 Generating Interrupt Vectors:

Requesting an interrupt by activating IR line causes Micro-78 program control to jump to location 70 (octal). The scheme proposed for identifying the interrupting device has to be a combined hard/soft were feature to avoid polling. The following is to be achieved by the CCC.

1. Generate an IR whenever any request from any of the interfaces comes along.

2. By executing 1/2 instructions at 70(octal) Micro-78 should be able to determine the source.

The heart of this part of the system is a priority encoder (Fig. 4.4). The output of this changes the contents of two hardware locations implemented by strip switches. Also one of its special outputs go low whenever any request is present at its inputs. In case of simultaneous requests the higher numbered input gets encoded. This then causes an identifying bit combination to be present in the locations mentioned (Two of the topmost locations) identifying the source.

The next step is to make the Micro-78 control jump to this unique location which is achieved as follows.

The following program made resident at 70(octal) would achieve the part (2) of the function as shown alongside

70 LHLD 177776 (octal) | (L) $\leftarrow$ (177776) & (H) $\leftarrow$ (177776+1)

73 PCHL (PCH) $\leftarrow$ (H); | (PCL) $\leftarrow$ (L)

where H & L refer to two registers available inside Micro-78 and PCH and PCL refer to upper and lower bytes of the PC (program counter).

The actual data transfer via data bus through Tri state buffers is exactly same as with other addressable locations. This is discussed in the next chapter which takes up the design of the SI.

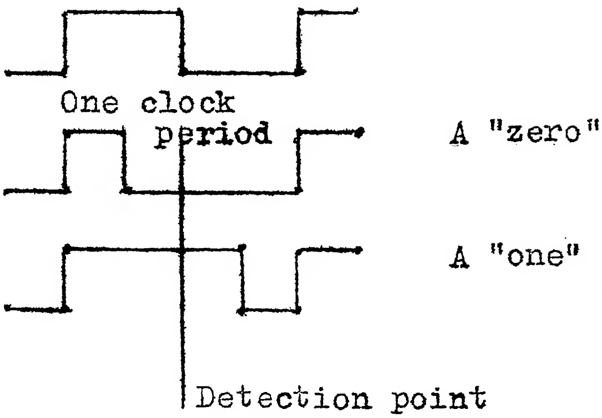


Fig. 4.1: Coding Scheme

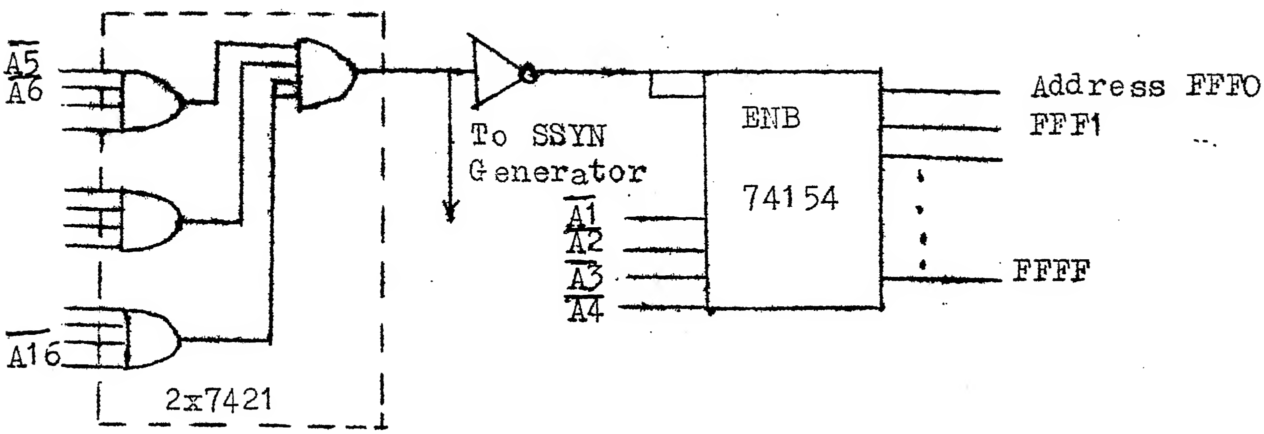


Fig. 4.2: Address Decoding Scheme

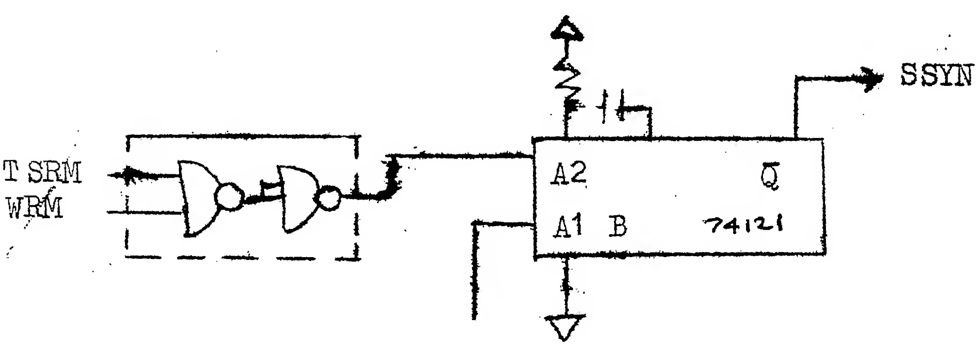


Fig. 4.3: SSYN Generation

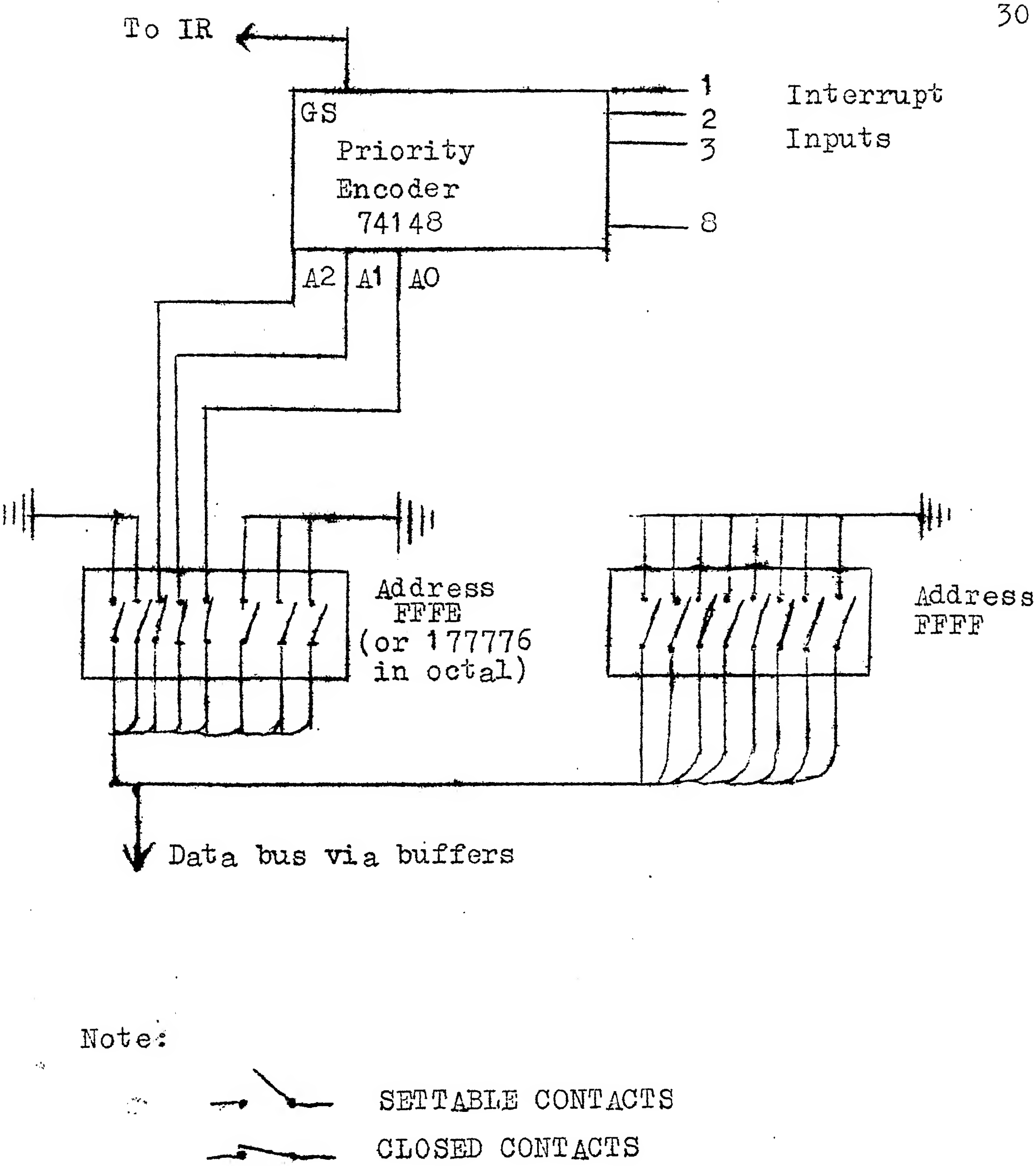


Fig. 4.4: Interrupt Vector Generation

CHAPTER 5

HARDWARE IMPLEMENTATION OF SENDING INTERFACE

This chapter discusses the implementation of the protocol as it applies to a sending interface. The protocol will be described in Hardware Design Language (HDL) CHU ; so that direct correspondence between the description and actual hardware can be established.

The Micro-78 has to communicate with two SI and two RI's (as of now). Each of these interfaces has two addressable registers. One register holds control and status information and is called control and status register (CSR). The other is the data register (DTRG) which contains the data loaded by CPU (in a SI) or loaded by the interface (in a RI) to be transferred into main memory.

The sending interface action can be summarized as follows. Fig. 5.1 gives the overall block diagram of the SI.

1. With power on CSR bits are cleared, SI sends SYN characters over the line. (This is the idling state).

2. Processor deposits the firxt byte of the message to be transmitted in DTRG and sets flags for initiation of message transfer (INIT) and byte handed over (TX) in the CSR.

3. At the end of current byte transmission, interface checks if at least one SYN has been sent after the last message was transmitted. If so it loads DTRG contents into parallel to serial converter register (TXRG) whose contents can be shifted out to the line.

4. A 1 bit odd parity is generated and appended to the byte in TXRG.

5. Data is shifted out serially MSB first onto the line.

6. As soon as data is transferred from DTRG to TXRG an interrupt is generated. The CPU transfers the next byte onto DTRG.

7. With the interrupt request following last byte transmission the CPU resets all CSR bits thereby causing SI to go back to step 1 after the current (last) byte has been transmitted.

5.1 Hardware Description of the SI:

The CSR is a 5 bit register implemented using D flip flops (3x7474; one $\frac{1}{2}$ 7474 unused). The CSR format is as shown in Fig. 5.2. The sending interface protocol in HDL follows.

Definitions:

Register DTRG (0-7), CSR (0-7), LC

Shift Register TXRG (0-7)

Clock PP

Character SYN (16 in HEX)

/($\overline{\text{INIT}}$ + $\overline{\text{MODE}}$).(LC=0).PP/TXRG $\leftarrow$SYN, MODE$\leftarrow$1 (S1)

/(LC$\neq$0).PP/Shift right TXRG, decrement LC (S2)

/INIT.MODE,(LC=0).PP/TXRG $\leftarrow$DTRG, TX $\leftarrow$ 0 (interrupts processor) (S3)

/ INIT.MODE. $\overline{\text{TX}}$.(LC=0).PP/ERR$\leftarrow$1(interrupts processor) (S4)

The combinations enclosed in // define various combinations of the CSR bits and other conditions. The action following it is the action to be taken when such condition is satisfied. The figures on the right hand column signify control signal no (S1,S2 etc.) which correspond to these condition. This has been resorted to for easy description.

5.2 Brief Explanation:

The CSR bits as depicted in Fig. 5.2 have the following significance.

INIT: The initiallization bit. It is set by processor (Pc) to indicate a message is ready to be transmitted. Reset when last byte has been transmitted.

MODE : This is set by interface to indicate at least one SYN has been transmitted. Reset at the end of message transmission by Pc to enable transmission of one SYN character over the line before next message starts.

TX : This is the bit which keeps track of bytes as they are handed over. Pc sets it when a byte is dumped into DTRG while SI resets it when the byte has been transferred to TXRG: This resetting is done at the trailing edge of S3 so that S4 is not activated during the same LC=0 cycle which will be incorrect.

INTF: or the interrupt flag. This is set whenever INIT is set and TX is reset, indicating interrupt condition. Setting of this or ERR generates an interrupt and this flag is used to indicate to the processor the nature of the interrupt.

ERR : If with previous byte transfer TX was reset and the interrupt thus generated was somehow ignored by the Pc; then the TX would remain reset at the current LC=0 cycle. This means DTRG contents are same as the old value since no new byte has been deposited into it. This is an error condition causing repetition of a byte and is flagged by this byte. Reset by Pc.

The action briefly then is that Pc sets INIT and TX after it has deposited the first byte of the scheduled message into the SI. SI finds INIT, MODE and TX set at the next LC=0 cycle and transfers 8 bit data to TXRG. It generates an `dd parity bit on these 8 bits and loads into a D flip flop appended to the TXRG. These 9 bits are shifted out. Meanwhile interrupt generated due to resetting of TX during transfer (TXRG DTRG) causes Pc to handover next byte and set TX. At the end of transmission Pc resets all bits in CSR. SI goes back to idling state.

5.3 SI Block Diagrams:

The hardware consists of the following functional blocks.

a) Clock generator

b) Data register and parallel/serial converter

c) Control signals

d) Coding circuit and line driver

e) CSR

The following sections explain the block diagrams in the above order.

5.3.1 Clock Generator: [Fig. 5.3]

The clock generator uses a 2 MHz crystal. Motorola's LM375 generates the basic clock which is divided down by

a chain of 7490's to get 40 KHz frequency. A final stage of 7474 generates 20 KHz (B) and 10 KHz (A). These two outputs are necessary for the encoder as discussed later.

5.3.2 Data Reg & parallel/serial converter [Fig. 5.4]

As mentioned in Chapter 4 the actual connection of data reg inputs to the data bus will be discussed here. Identical arrangements are used every where registers connect to data bus. Only difference being that the enabling signal for the buffers is derived from read/write signal depending on the read/write commands to from CPU for which data is to be given out or taken in. The buffers used are unidirectional and each chip contains four buffers.

The DTRG outputs and stripswitch with SYN combination both terminate on the Mux. Except when S3 is active, SYN combination is selected. Mux output goes to parity gen input as well as the TXRG input. As per the hardware algorithm the TXRG and the 9th bit ($\frac{1}{2}$ 7474) are to be loaded whenever S1+S3 is active. PP does the shifting out. To avoid ambiguity in loading data S1+S3 should be slightly delayed compared to the switch over leading edge of S3. This gets achieved as deriving S1+S3 needs an extra gate to realize it Fig. 5.5 .

5.3.3 Control Signals [Fig. 5.6]

Control signals are all low active since the signals to

set/reset 7474's are to be low signals. Further the reason for using the asynchronous inputs of the 7474 was that one would need extra logic to derive properly timed clock had the D & CLK inputs been used. On the other hand when CPU communicates the write pulse would come in proper sequence. If at all any delay is required, it will be for this signal alone. It is also evident that there would not be any race conditions as the two types of changing CSR bits occur at predetermined times when no such condition prevail.

Actual generation of the signals are self explanatory.

5.3.4 Coding Circuit and Line driver [Fig. 5.7]

Coding scheme was discussed in the previous chapter. It was mentioned there that they are sent inverted onto the line. Reason for this was that initially PLL was proposed to be used. MC 4044 and MC 4024 pair constitutes such a loop. The phase detector is sensitive to negative transitions so that for proper locking it was necessary to have negative transition at the beginning of a bit period. It was found NSN-79 , Section 4.6 the voltage controlled multivibrator in this set was very voltage sensitive. Particularly for Micro-78 interfaces (3 RI's in all) this would become a trouble some affair and was given up in favour of a monostable circuit although the encoding scheme was retained.

The coding logic could easily be derived as follows

Serial output = I.A + I.B + A.B

where I = data input from serial output of TXRG

A = 10 KHz basic clock

B = 20 KHz clock

The implementation is shown in Fig. 5.6. The common emitter output of the NPN transistor provides impedance matching and the current drive. No serious distortion in pulse shape was observed while driving 62 A/U 93 ohms cable of upto 300 ft length.

5.3.5 <u>The CSR controls</u> [Fig. 5.8]

The reasons for choosing the asynchronous and synchronous control inputs of the 7474's of the CSR for control by CPU and that by the interface have already been discussed in Section 5.3.3.

CPU communication is similar to that discussed for DTRG. The end signal is derived the same way as there. Since the same edge is to be used for triggering in the data, CLK signals are derived as shown.

The interface controls the bits as shown. All the clear inputs except tht of TX bit are connected together and

connected to master clear (MC) line of the Micro-78. This facilitates clearing of all bits when Micro-78 is started up. TX bit is cleared by the interface for every byte transfer. MC and delayed S3 are ored together for this purpose. The hardware algorithm discussed in Section 5.1 directly indicates why the particular signals S1, S2 etc go where they go.

The line counter (LC) is not a part of the CSR but it is an essential part of the signal generation. Any action by the interface occur at LC=0 points. Implementation of LC is shown along with CSR bits implementation. Every time count is completed ripple clock output causes the parallel input (a count of 9) to be loaded and then the count down starts. This serves to mark out the 9 bit periods by giving out positive going Max/Min output which is used as LC=0 pulse.

5.3.6 <u>Interrupt Generation</u> [Fig. 5.9]

Whenever the INTF or the ERR flag is up an interrupt request is to be transmitted to the CCC priority encoder circuit. Since one section of a 7400 was available, $\bar{Q}$ outputs of INTF and ERR were used to get the OR function. This request would be coded into proper outputs if another higher priority request is not present and an request made on the Micro-78 IR line.

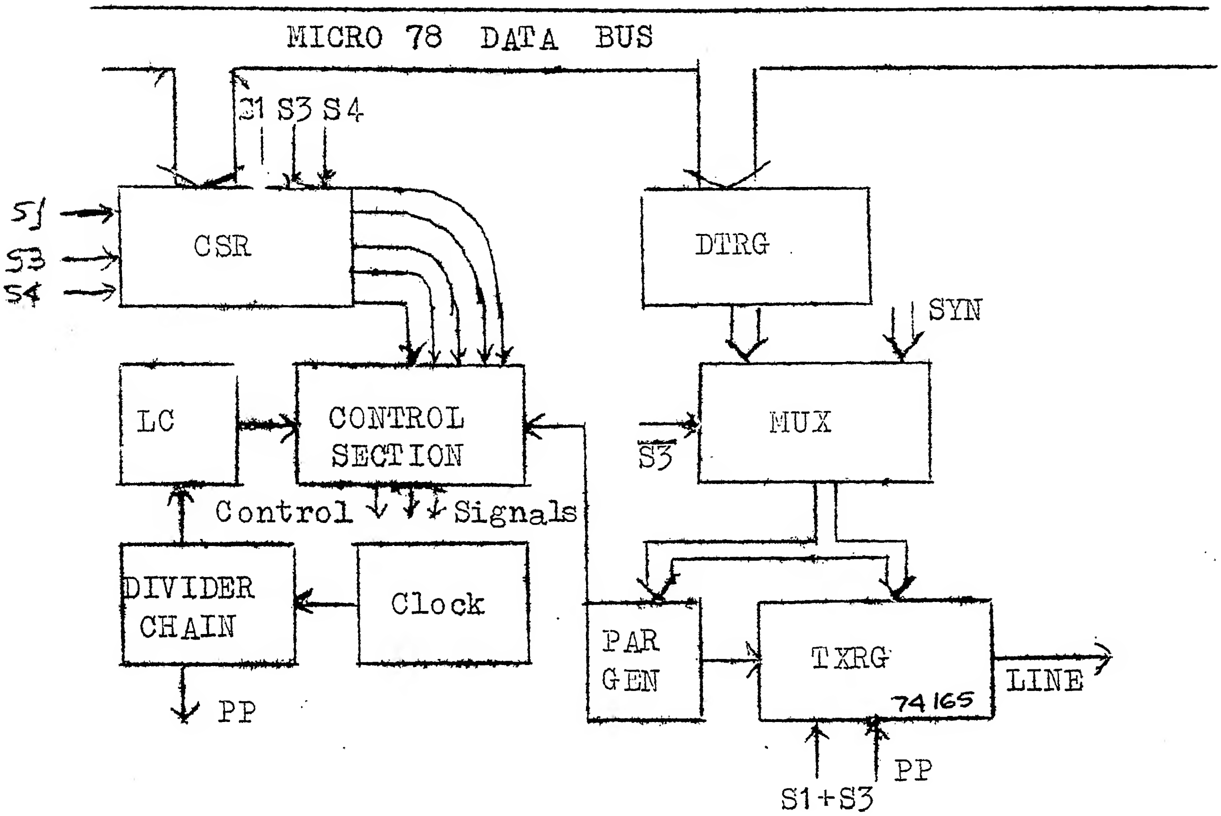


Fig. 5.1: Overall Block Diagram

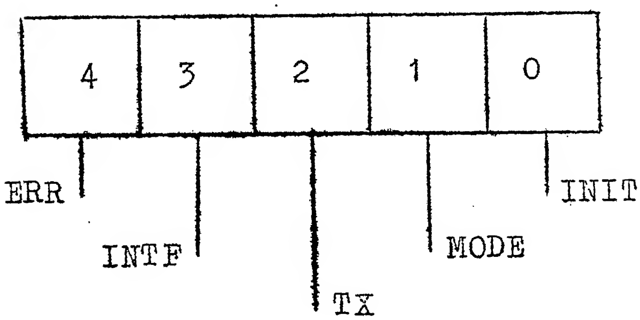


Fig. 5.2: The CSR Bits

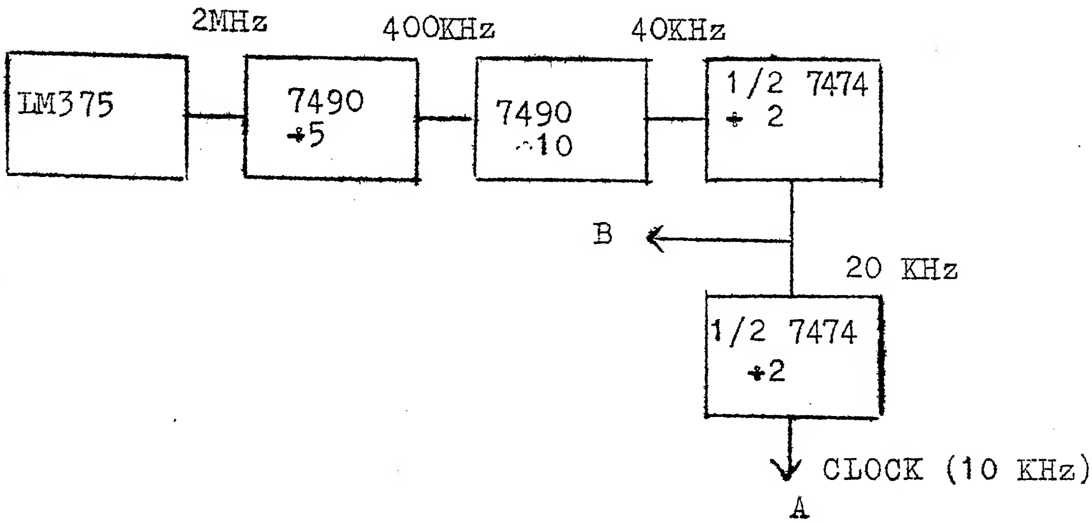


Fig. 5.3(a): Divider Chain

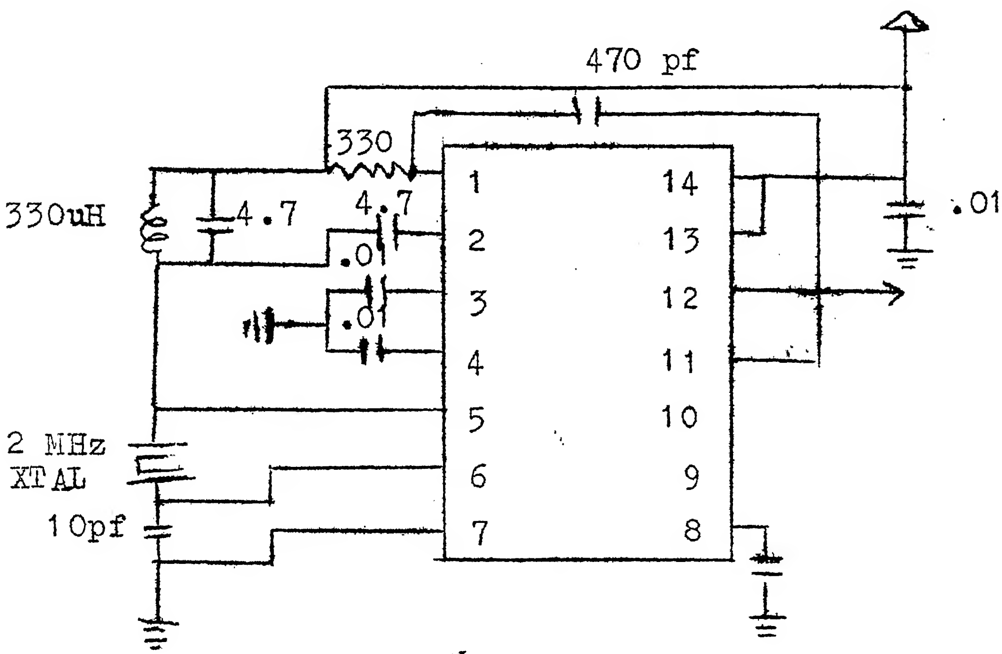
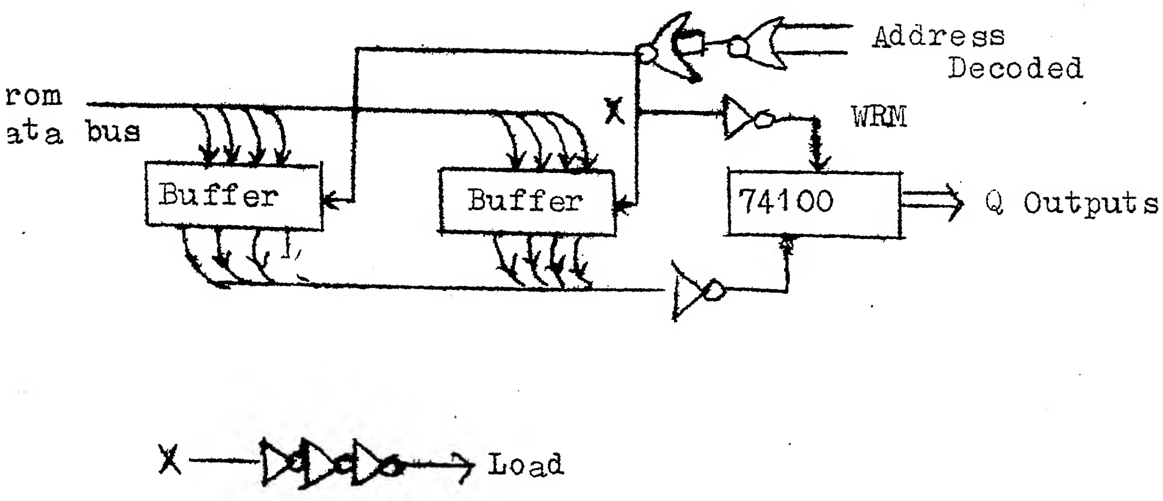
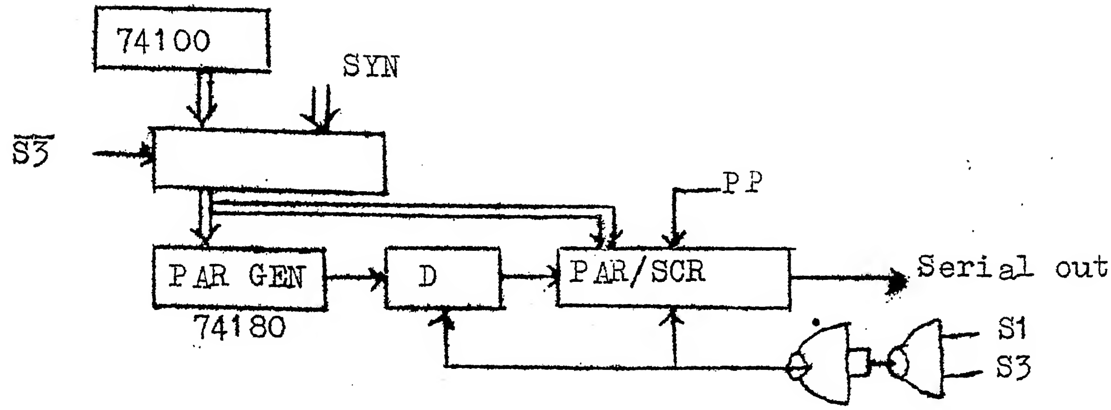


Fig. 5.3(b): Clock Generator



(a) Connection with data bus



(b) DTRG to TXRG Interconnections

Fig. 5.4: Data Reg & Parallel to Serial Register

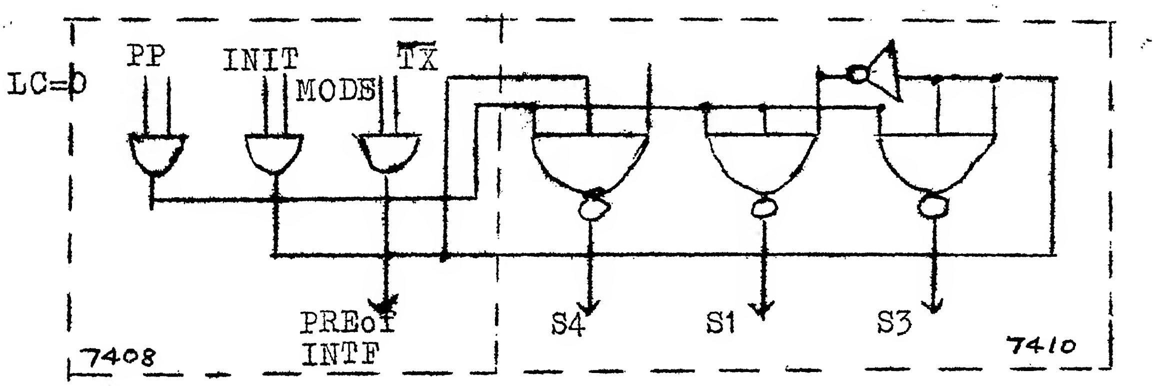


Fig. 5.6: Generation of Control Signals.

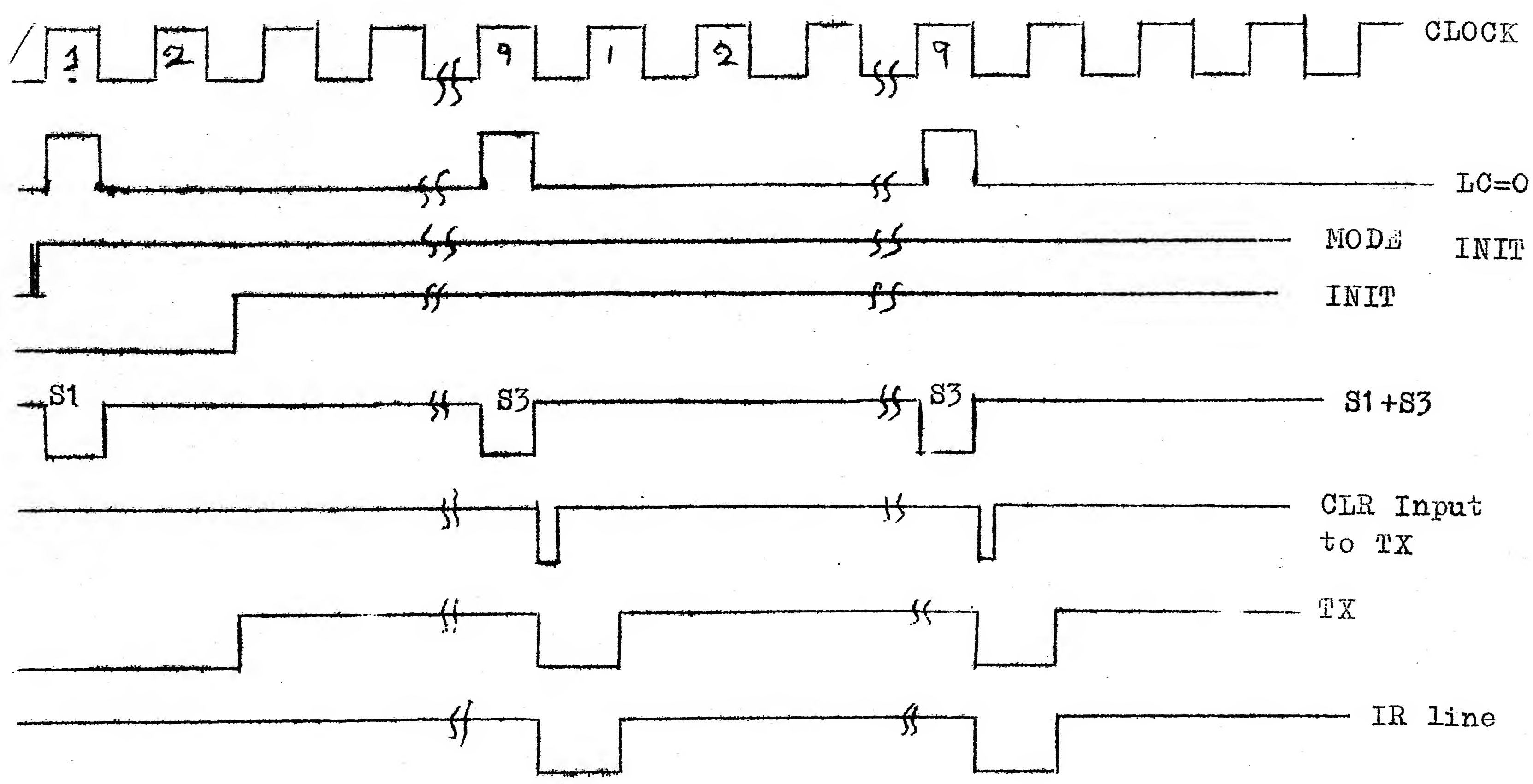


Fig. 5.5: Sending Interface Timing Diagram

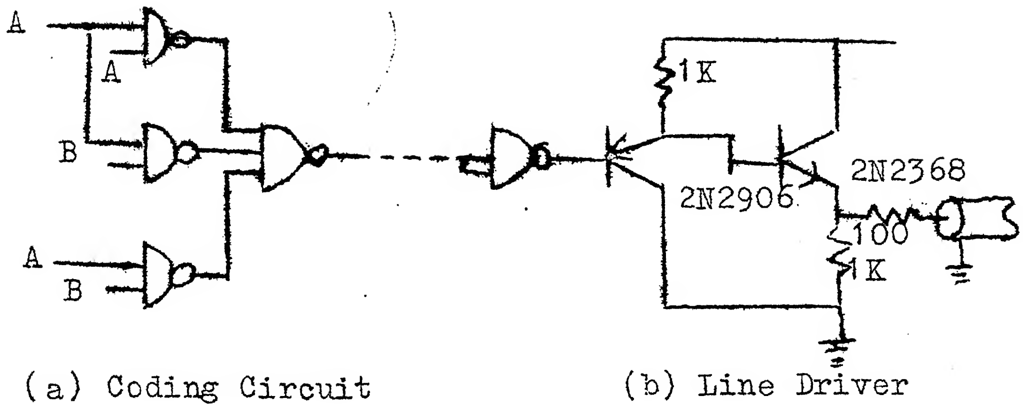


Fig. 5.7: Coding Circuit & Line Driver

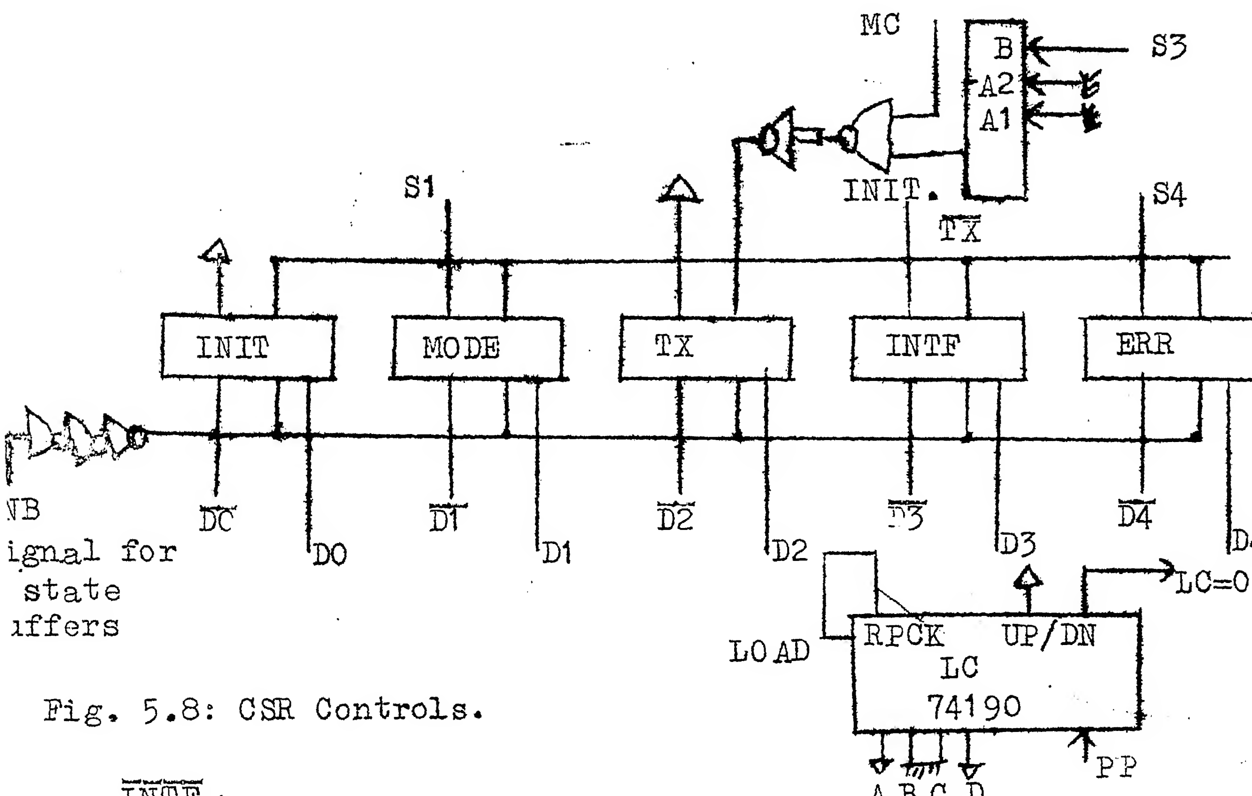


Fig. 5.8: CSR Controls.

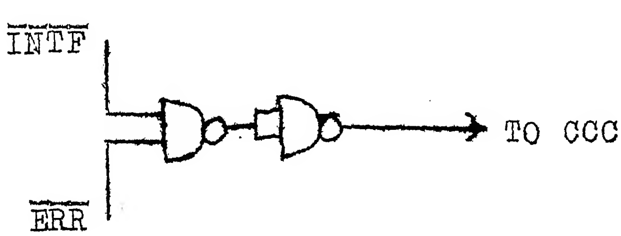


Fig. 5.9: Interrupt Generation.

CHAPTER 6

RECEIVING INTERFACE HARDWARE

The receiving interface has the following function.

i) To generate a clock synchronous to the transmitted clock.

ii) To mark out byte boundaries by detecting a SYN character over the line.

iii) To detect SOM/ACK in the idle state and interrupt the Pc.

iv) To interrupt Pc every time a byte is handed over to the DTRG in the message reception mode.

v) To check parity and set a flag if there is a failure.

vi) To set a flag indicating over run if any interrupt request is not serviced.

6.1 Hardware Description of the RI:

As in the Sending Interface Receiving Interface too has two addresseable registers the DTRG and the CSR. The overall block diagram is shown at Fig. 6.1. CSR format is shown in Fig. 6.2.

MODE : This is set by the interface to indicate a SOM/ACK has been received. Reset by Pc when the message is completely received when it is an SOM else immediately if it is an ACK.

RX : Byte handover flag: Set by interface when a byte has been received and handed over to the data register. Reset by the Pc when it is taken in.

INTF: Interrupt flag: Set when Mode & RX flags are up. This causes an interrupt request to be generated. Reset when the interrupt is serviced by the processor.

OVR : Overrun flag: Set by interface to indicate an overrun condition. This too causes an interrupt. This and the previous flag then helps in discriminating the cause for interrupt, cleared by Pc.

PE : Parity Error flag: Set by interface when parity is in error for the current byte. Cleared by processor.

SF : Byte Synchronization flag: Set and cleared by the interface. Set when a SYN is recognized first. It is reset after byte synchronization has been achieved but incoming byte is not SOM/ACK to indicate start of message and not even a SYN to indicate transmitter is idling. This being an indication of something being wrong, interface starts looking for a valid SYN to achieve byte synchronization.

The hardware algorithm for the receiving interface is as follows:

$/\overline{SF}.PP//$ If RXRG=SYN then SF$\leftarrow$1, LC$\leftarrow$9 (S1)

$/PP//$ Shift right RXRG, RXRG(8)$\leftarrow$(bit from the line),
decrement LC (S2)

$/SF.\overline{MODE}.(LC=0).PP/$If RXRG = SOM/ACK then (S3)

MODE $\leftarrow$1, DTRG$\leftarrow$RXRG, RX$\leftarrow$1

else if RXRG $\neq$ SYN then SF$\leftarrow$0 (S4)

$/SF.MODE.(LC=0).\overline{RX}.PP/$ DTRG$\leftarrow$RXRG, RX$\leftarrow$1 (S5)

If parity error then PE$\leftarrow$1

$/SF.MODE.(LC=0).RX.PP/$ OVR$\leftarrow$1 (S6)

RXRG is the serial to parallel converter shift register that receives the data from the line. LC is the line counter. When SF is set LC is initialized to 9 and subsequently for all LC=0 too it is loaded with 9. Thus once set, subsequent LC=0 would mark out the byte boundaries.

The hardware design is similar to that of sending interface. The hardware consists of the following functional blocks.

a) Clock recovery circuit

b) Data reg and serial/parallel converter

c) SOM, ACK, SYN detectors

d) Control signal generation

e) CSR.

The following sections explain the block diagram in the above order.

6.2 Cloc'- Recovery Circuit 'ig. 6.3 :

Clock is recovered as shown and follows the method discussed in Section 4.1.

6.3 Data Register and the RXRG Fig. 6.4 :

Data from the line gets clocked into the D flip-flop at the mid bit positive transition. This D output gets shifted into the serial to parallel converter so that at the end of 9 bit periods one byte of data is in the RXRG followed by the parity bit in the D flip-flop. The parity checker (74180) uses this bit to determine if there is a parity failure. The data is transferred into the DTRG whenever S3 or S5 is active.

6.4 SYN, SOM and ACK detectors Fig. 6.5 :

Three 3 line to 8 line detectors are used for this purpose. These three characters in octal are 026, 001 and 174 respectively. The RXRG outputs are connected to the inputs of the detectors as shown. A is the LSB. Corresponding output is low when a particular bit combination is present at the input.

6.5 Control Signal Generation [Fig. 6.6] :

Fig. 6.5(a) shows how the character detected signals are generated. The 3 input NOR gates are implemented by 7427's. The detected outputs are positive going while the control signals like S1 are low going. Reasons for this is the same as discussed in the previous chapter. S1 is the byte synchronization signal. Immediately a SYN is detected S1 makes LC load 9 into itself. From then onwards the ripple clock output makes it load 9 at the end of 9 bit count down so that the max/min output marks out the proper byte boundaries. Fig. 6.5(b) gives the details of it.

Fig. 6.5(c) details the generation of control signals as shown in the hardware description. A number of inverters have been used in the control signal generation. This was necessary to get the control signals as low active. Gate delays involved are not critical compared to a clock period of 100 sec, the period of the LC=0 pulse.

Fig. 6.5(d) shows how the setting of INTF is done and the signal for setting RX is generated. Reasons for using a monostable are the same as for the sending interface TX resetting.

6.6 The CSR Controls [Fig. 6.7]:

Like the sending interface CSR the design philosophy is the same here. Since SF need not be accessible by the

CPU no writing facility is given for this bit. However it can be read by Pc. $\bar{Q}$ outputs are used for this purpose and they go via 3 state buffers. Rest of the connections correspond exacely to the hardware description.

6.7 Interrupt Generation [Fig. 6.8] :

As in the sending interface, there are two interrupt sources in the receiving interface so that interrupt function is INTF + OVR. This is implemented as shown.

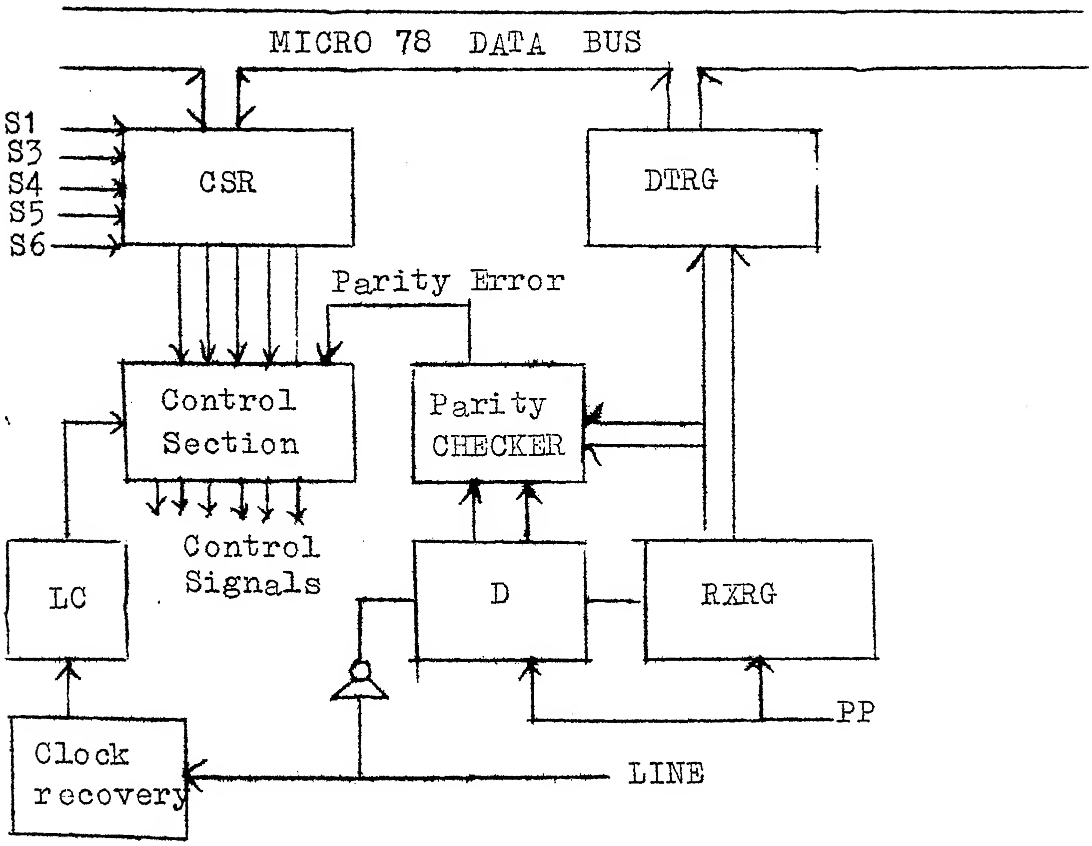


Fig. 6.1 : Overall Block Diagram

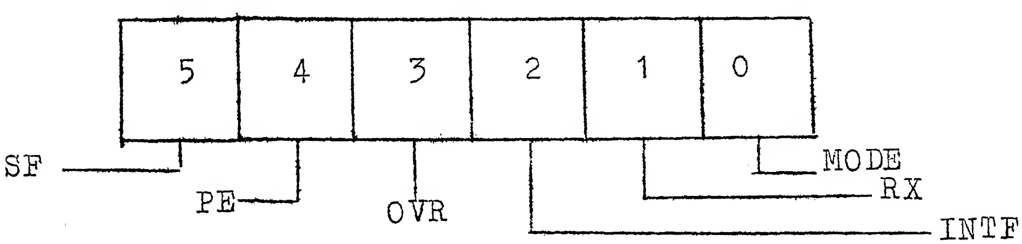


Fig. 6.2: CSR Format

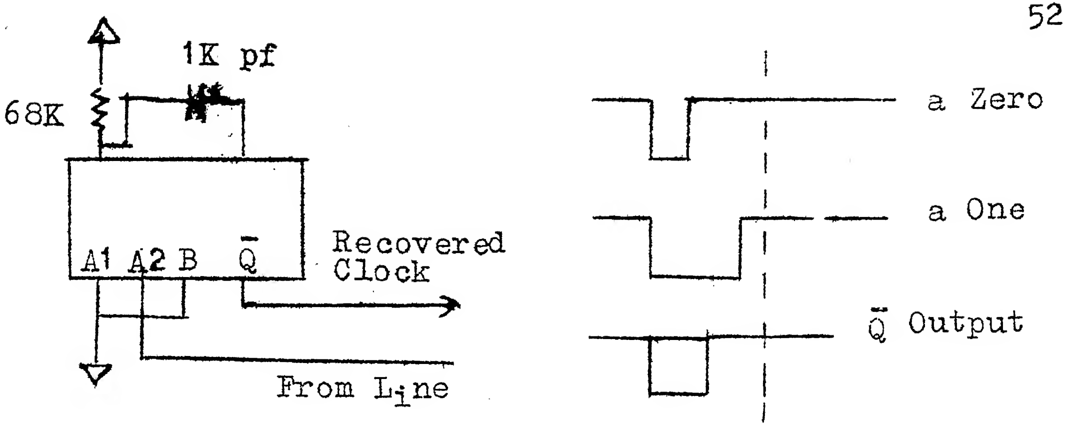


Fig. 6.3: Clock Recovery

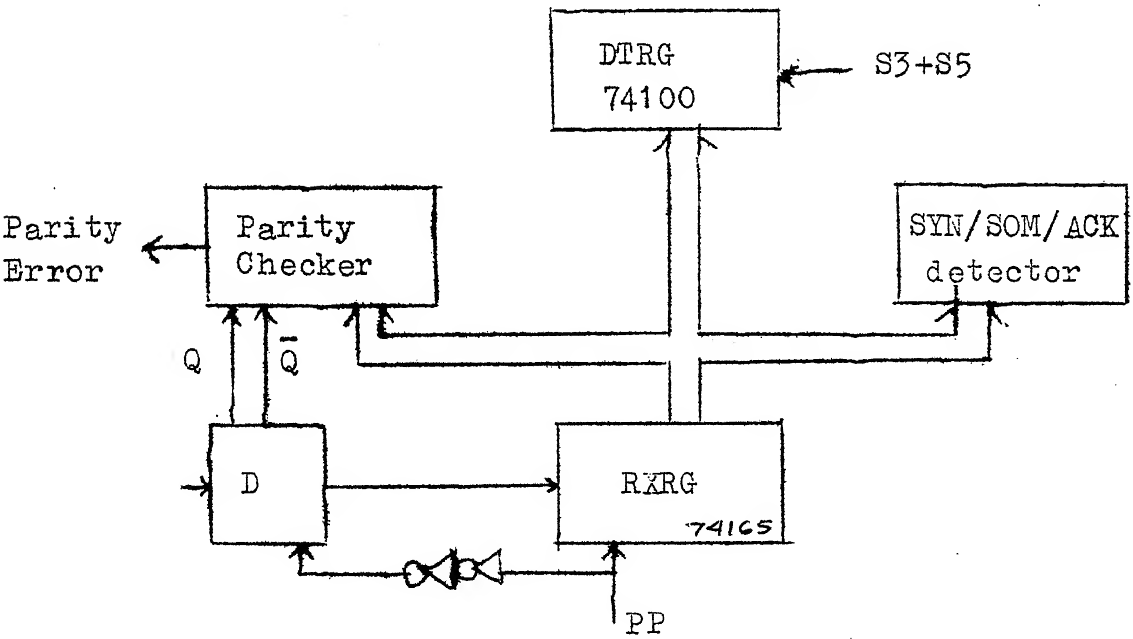
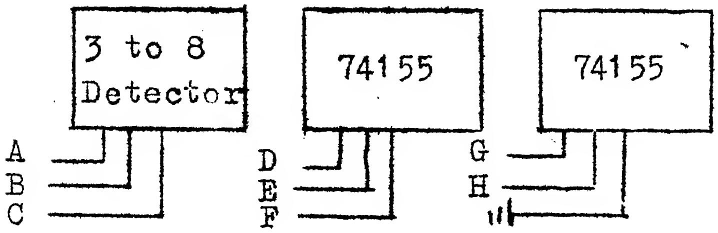
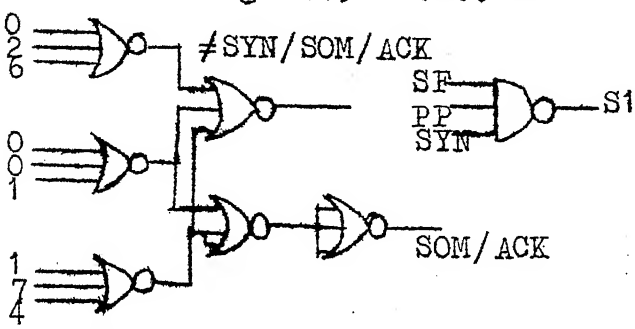


Fig. 6.4: DTRG and RXRG Connections

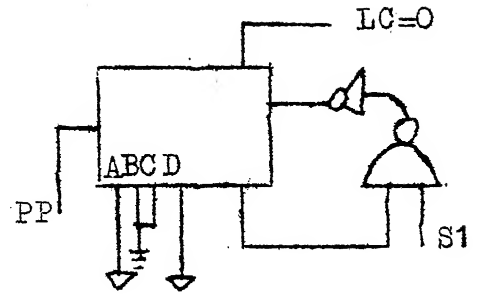


From RXRG

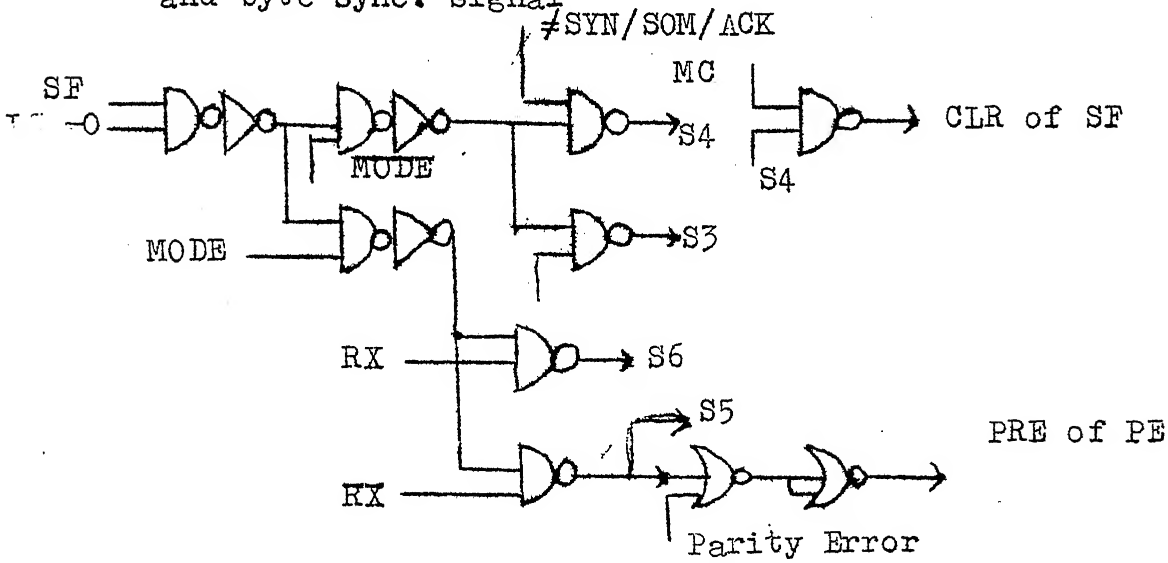
Fig. 6.5: Detectors



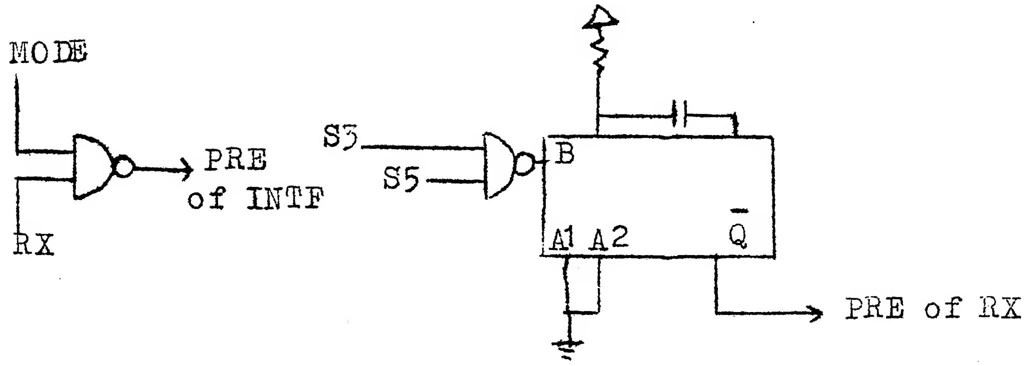
(a) Character detected signals and byte sync. signal



(b) Line counter controls



(c) Generation of Control Signals



(d) INTF & RX setting

Fig. 6.6: Control Section Details.

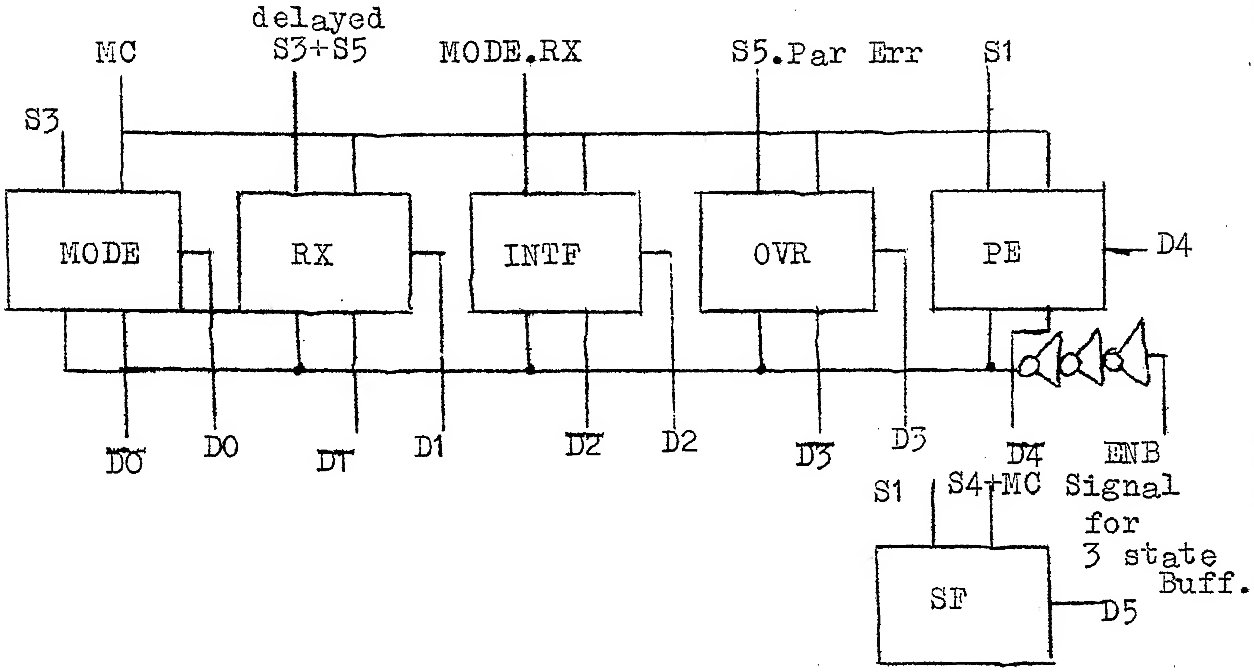


Fig. 6.7: CSR Control

Fig. 6.8: Interrupt Generation.

CHAPTER 7

CONCLUSIONS

The aim of the project was to link up TDC-316, IBM 1800 and DEC 1090 systems in a star network with Micro-78 as the switching centre. Work on the Micro-78 - DEC 1090 interface could be started only very recently since DECNET protocol details were received very late. Also, the Micro-78 to be used as the switching point has not arrived yet. So initial testing of all the three interfaces for TDC 316, IBM 1800 and Micro-78 was done by looping back SI to its own RI.

The testing of Micro-78 interfaces was done at the ACES. All bit combinations from 0 to 377 (octal) were sent and received back without errors. Since the micro-78 computers does not have the programme timer features, the time out function and outomatic retransmission functions could not be checked.

On looking back over the system design work the following are worth noticing:

a) The protocol developed treats SOM or ACK identically. Different actions due to SOM or ACK are taken by the Pc after identifying ACK or SOM

through software. With an acknowledgement in the same format as a normal message, ACK need not be recognized at all.

b) Data rates cannot be easily changed. Hardware modifications are necessary to achieve this. On the receiving side it is much easier to achieve as already discussed.

Future work can possibly proceed on the following directions:

1) Hardware development of DMA controllers for Micro-78 and TDC 316 could be taken up. It should be possible to go to much higher data rates this way.

2) It should be possible to modify the present hardware to form a data loop. This would be complicated by the fact that there is provision only for two synchronous lines on the DEC10 side. A direct link between a SI of a Pc with its own RI would be required. Further, decoders to identify source and destination addresses in a message would be required.

A lot of software development needs to be done before the network really gets going. Higher level protocols

and software packages to implement them would be the first step. Complete software system needs to be implemented in the Micro-78 to achieve the function as a central switch.

Using Micro-78 microcomputers as NODES for the three Hosts, the communication overhead could be completely turned over to these communication processors. These would serve as points to extend the net work further.

REFERENCES

[ABRA and KUO] Abramson, N. and Kuo, F.F. (eds): Computer Communication Networks (1973).

[BRAN-72] Brandt, G.J. and Chretien, G.J.: Methods to Control and Operate a Message Switching Network in Proceedings of the Symposium on Computer Communications Networks and Teletraffic (1972), 263-276.

[CHU] Chu, Yaohan : 'Computer Organisation and Microprogramming', Prentice-Hall, 1972.

[CROW-75] Crowther, W.R., et al: Issues in Packet Switching Network Design: Proc. AFIPS NCC 44(1975), 161-175.

[CVK-79] C.V. Krishna, :IIT/K Computer Network -IBM-1800 Hardware Interface: M.Tech. Thesis.

[DAVI-68] Davies, D.W.: 'The Principles of a Data Communication Network for Computers and Peripherals' Advances in Computer Communications (1974) Edited by Chu, W.W.

[DAVI & BARB] Communication Networks for Computers (1973), Davies, D.W., and Barber, D.L.A.

[FRAN-72] Frank, H., and Chou, W.: Topological Optimisation of Computer Networks: Proc. IEEE 60, 11 (Nov. 1972), 1385-1397.

[FRAS-76] Fraser, A.G.: The Present Status and Future Trends in Computer Communication Technology: Computer (Sept.,1976), 10-19.

[GRAY-72] Gray, J.P.: Line Control Procedures: Proc. IEEE 60, 11 (Nov. 1972), 1301-1312.

[KAHN-72] Kahn, R.E.: Resource Sharing Computer Communications Networks: Proc. IEEE 60, 11 (Nov.1972), 1397-1407.

[KLEI] Kleinrock, L.: Queuing Systems, Volume II, Computer Applications (1976).

[NSN-79] NS Narayanan: IIT/K Computer Network - TDC-316 Hardware Interface: M.Tech. Thesis.

[ORNS-72] Ornstein, S.M., et al: The Terminal IMP for the ARPA Computer Network: Proc. AFIPS SJCC 40(1972), 243-254.

[TTLH] Texas Instruments Inc: 'Designing with TTL Integrated Circuits', ISE, McGraw-Hill,1971.

[WALD-75] Walden, D.C.: Experience in Building, Operating, and Using the ARPA Network: Second USA - Japan Comp. Conf., Tokyo (Aug. 1975).